In-network Principal Component Analysis with Diffusion Strategies

Nisrine Ghadban, Paul Honeine, Farah Mourad-Chehade, Clovis Francis, Joumana Farah

In-network principal component analysis and diffusion strategies

Nisrine Ghadban, Paul Honeine, Farah Mourad-Chehade, Clovis Francis, Joumana Farah

Abstract

Principal component analysis is a very well-known statistical analysis technique. It determines principal axes that allow relevant data reduction. Thus, it is very useful in networks where a large number of measures is performed. However, such a technique needs high-complexity computation, which does not scales well in large networks. This paper proposes to overcome this issue, by describing several in-network strategies that estimate the principal axes with lower computational complexity. These strategies are either noncooperative or cooperative with information diffusion. The performance of the proposed strategies are illustrated on diverse applications, such as image processing and dimensionality reduction of time series in wireless sensor networks.

Index Terms

Principal component analysis, network, adaptive learning, distributed processing, dimensionality reduction.

I. Introduction

Principal component analysis (PCA) is one of the most popular techniques for data analysis and processing, investigated in statistical analysis, data compression, and feature extraction [1]. It consists in determining a subspace that retains the largest variance of the data. This subspace is spanned by the most relevant principal axes. Projecting the data onto the resulting subspace leads to the so-called *principal components*. By representing the data by these principal components, one reduces the dimensionality and extracts the data structure. The PCA technique is very useful in many applications involving multivariate analysis, e.g., data validation, fault detection [2], and quality control [3]. It provides a powerful tool for data analysis and pattern recognition which is often used in signal and image processing [4]. Moreover, in the context of sensor networks, the PCA has been investigated to extract features from noisy samples [5], compress and denoise time series measurements [6], [7], as well as for detection of intrusion [8] or anomaly [9].

The conventional PCA requires the eigen-decomposition of the sample covariance matrix, where the eigenvectors associated to the largest eigenvalues are the most relevant principal axes. Therefore, it is necessary to send all data to a fusion center (FC), where the covariance matrix is estimated and eigen-decomposed. Such configuration is not scalable and inappropriate for network applications. Moreover, the computational complexity of such eigen-decomposition is cubic with the size of the dataset. Several attempts have been made to alleviate this problem, such as in [10], [11] where algorithms are investigated to compute the eigenspace, but still with high computational cost. More recently, in [12], a technique was proposed in which the FC only receives the principal components, instead of the whole time series. In [13], the most relevant principal axis is estimated by the power iteration method. This method requires the computation of the sample covariance matrix, making it inappropriate for in-network processing.

In this paper, we propose to estimate the most relevant principal axes without computing the sample covariance matrix. To this end, we investigate the Oja's neural-based rule [14], which has been recently revisited in [4] for nonlinear PCA with kernel-based machines. We extend our study in [15] to incorporate several cost functions, including the information theoretical criterion derived in [16], Rayleigh quotient, LUO and OJAn cost functions derived in [17], [18]. Within a network setting, several in-network strategies are proposed, including noncooperative

N. Ghadban, P. Honeine and F. Mourad-Chehade are with the Institut Charles Delaunay (CNRS), Université de technologie de Troyes, France.

N. Ghadban is also with the Faculté de Génie, Université Libanaise, Lebanon.

C. Francis is with Faculté de Génie, Université Libanaise, Lebanon.

J. Farah is with the Telecommunications department, Faculty of Engineering, Holy-Spirit University of Kaslik, Lebanon.

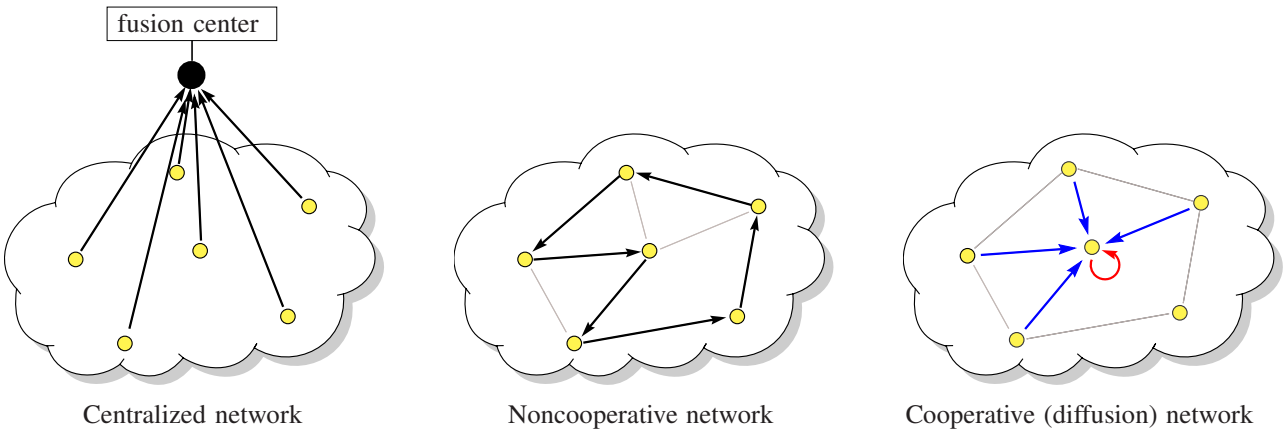

Fig. 1. Illustration of information processing in a network: While the centralized network requires a fusion center, the noncooperative and cooperative networks provide in-network processing. The noncooperative network process the information by using a given routing path. The cooperation network operates information diffusion, using either a combine-then-adapt or an adapt-then-combine strategy.

and cooperative strategies. In the latter, sensors cooperate with each other by information diffusion in order to estimate the principal axes. We derive two diffusion strategies, the so-called combine-then-adapt and adapt-then-combine strategies, which have been recently investigated in linear adaptive filtering literature by A. Sayed *et al.* in [19]. To the best of our knowledge, our study is the first work that investigates these adaptation strategies for unsupervised learning, as given here with the PCA. To provide a comprehensive presentation throughout this paper, we first study in detail the estimation of a single principal axis for a given cost function, before generalizing it to different cost functions. We then extend the proposed approach to the estimation of multiple principal axes, by investigating orthogonalization processes, such as the well-known Gram-Schmidt orthogonalization and the deflation scheme.

The rest of this paper is organized as follows. Section II briefly introduces the network topology. We describe the proposed strategies for estimating the first principal axis based on Oja's rule in Section III. In Section IV, we generalize the proposed approach to other cost functions, beyond Oja's rule. Section V extends the strategies for multiple axes extraction. In Section VI, we present algorithms well-known in the literature. Section VII provides experimentation results and discussions, whereas Section VIII concludes the paper.

II. Network setting

Consider a network of N connected nodes (also called *agents* in the literature). In such a network, any two nodes are necessarily linked: either directly if they are neighbors, or through other intermediate nodes. On the other hand, there are mainly two types of networks: centralized and decentralized networks. In the former, the nodes are connected to a FC, and in the latter the nodes are connected in a noncooperative way by a given routing path, or in a cooperative way, where each node communicates with its neighbors. We denote by $\mathcal{V}_k$ the set of indices of the neighboring nodes to node k, with $k = 1, ..., N$, *i.e.*, the nodes that are directly connected to it. We consider that the node k is adjacent to itself, that is to say $k \in \mathcal{V}_k$. The network topologies studied in this paper are illustrated in Fig. 1.

We are interested in the case where the nodes estimate the same principal axis $\boldsymbol{w}$ based on some measured information corresponding to a common phenomenon. Let $\boldsymbol{x}_k$ be the p-by-1 vector collected by the node k, for $k = 1, \ldots, N$, p being the measurements' dimension, and let $\mathbb{X} \subset \mathbb{R}^p$ be the space of these collected data (assumed to be zero-mean), with the conventional inner product $\boldsymbol{x}_k^\top \boldsymbol{x}_l$ for any $\boldsymbol{x}_k, \boldsymbol{x}_l \in \mathbb{X}$. We denote by the scalar value $y_{\boldsymbol{w},k} = \boldsymbol{w}^\top \boldsymbol{x}_k$ the inner product associated with the orthogonal projection of any $\boldsymbol{x}_k \in \mathbb{X}$ onto the vector $\boldsymbol{w} \in \mathbb{X}$ and by $y_{\boldsymbol{w}}$ the scalar random variable taking the values $y_{\boldsymbol{w},k}$, with $k = 1, ..., N$. The ultimate goal would be to compute $\boldsymbol{w}$, optimaly for some given cost function and according to the network topologies, to keep afterwards only $y_{\boldsymbol{w},k}$ and $\boldsymbol{w}$, for $k = 1, ..., N$, allowing us to represent efficiently the data $\boldsymbol{x}_k$ when needed.

III. In-network PCA

In this section, we describe the different strategies depending on the network topology. We first consider the centralized network and expose the conventional strategy for PCA. We then derive our new strategies applied to the decentralized networks for extracting the first principal axis. We extend these techniques to the case of multiple principal axes in Section V.

A. Centralized strategy

Here, the nodes are assumed to be connected to a FC. The latter receives the collected data without any local processing from the nodes. In order to extract the first principal axis, the FC maximizes the variance of the projected data (assumed to be zero-mean), namely $\max_{\boldsymbol{w}} \mathbb{E}(y_{\boldsymbol{w}}^2)$, where $\mathbb{E}(\cdot)$ is the expectation over the density of the input data. By taking the empirical estimation of the variance of the projected data with respect to the available samples, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, we get $\max_{\boldsymbol{w}} \boldsymbol{w}^\top \boldsymbol{C}\boldsymbol{w}$, where $\boldsymbol{C} = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_k \boldsymbol{x}_k^\top$ is the covariance of the measured data. The problem takes the form $\boldsymbol{C}\boldsymbol{w} = \lambda \boldsymbol{w}$. This is the well-known eigen-decomposition problem where $\boldsymbol{w}$ is the eigenvector associated with the eigenvalue λ of $\boldsymbol{C}$. In fact, the latter is the corresponding variance of the projected data, since

$$\boldsymbol{w}^\top \boldsymbol{C}\boldsymbol{w} = \boldsymbol{w}^\top \lambda \boldsymbol{w} = \lambda \boldsymbol{w}^\top \boldsymbol{w} = \lambda.$$

Therefore, we must consider the eigenvector associated with the largest eigenvalue to maximize this variance. Note that the k-th principal component of $\boldsymbol{x}$ is $\boldsymbol{w}_k^\top \boldsymbol{x}$ and $\text{var}[\boldsymbol{w}_k^\top \boldsymbol{x}] = \lambda_k$, where the k-th principal axis $\boldsymbol{w}_k$ is the eigenvector corresponding to the k-th largest eigenvalue λ_k.

Having the data, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the FC solves the eigen-decomposition problem. The eigenvector associated with the largest eigenvalue corresponds to the principal axis, denoted $\boldsymbol{w}_*$ in the following. The computational complexity of the eigen-decomposition problem is $\mathcal{O}(p^3)$. There is also the communication complexity, which is $\mathcal{O}(Np)$ over a distance $\mathcal{O}(1)$. In the following, we propose strategies that avoid the computation of the sample covariance matrix and its eigen-decomposition, thus allowing to reduce significantly the computational complexity.

B. Noncooperative strategy

We consider an in-network scheme where the first principal axis is learned adaptively instead of directly maximizing the overall projected variance $\mathbb{E}(y_{\boldsymbol{w}}^2)$ and explicitly investigating the covariance matrix. To this end, we consider an "instantaneous" estimation of the principal axis $\boldsymbol{w}$ at each node. According to a given routing process, each node k receives an estimate $\boldsymbol{w}_{t-1}$ from another node, and adjusts it using its own data $\boldsymbol{x}_k$ by maximizing $y_{\boldsymbol{w},k}^2$. In other words, the node k minimizes the "instantaneous" quadratic reconstruction error, namely

$$J_k(\boldsymbol{w}) = \tfrac{1}{4}\|\boldsymbol{x}_k - y_{\boldsymbol{w},k}\boldsymbol{w}\|^2. \tag{1}$$

The gradient of this cost function with respect to $\boldsymbol{w}$ is

$$\nabla_{\boldsymbol{w}} J_k(\boldsymbol{w}) = y_{\boldsymbol{w},k}(y_{\boldsymbol{w},k}\boldsymbol{w} - \boldsymbol{x}_k), \tag{2}$$

which leads to the gradient descent formulation

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \eta_t \left(\boldsymbol{x}_k\, y_{\boldsymbol{w}_{t-1},k} - y_{\boldsymbol{w}_{t-1},k}^2\, \boldsymbol{w}_{t-1}\right), \tag{3}$$

where η_t is the learning rate. It turns out that this update rule is essentially Oja's rule [14], which is a single-neuron special case of the generalized Hebbian learning [20], applied here within the in-network setting described above.

This learning rule converges to an equilibrium state, which is the first principal axis $\boldsymbol{w}_*$. In fact, when $\boldsymbol{w}_t$ converges to some state $\boldsymbol{w}$, we have from (3) that $\boldsymbol{x}_k y_{\boldsymbol{w},k} = y_{\boldsymbol{w},k}^2 \boldsymbol{w}$, namely $\boldsymbol{x}_k \boldsymbol{x}_k^\top \boldsymbol{w} = y_{\boldsymbol{w},k}^2 \boldsymbol{w}$. Averaging the latter expression over the whole data, we get the well-known eigen-decomposition problem of the covariance matrix $\boldsymbol{C}\boldsymbol{w} = \mathbb{E}(y_{\boldsymbol{w}}^2)\, \boldsymbol{w}$, where the eigenvalue is the squared output $y_{\boldsymbol{w},k}$ to be maximized. Therefore, the update rule (3) converges to the eigenvector associated with the largest eigenvalue of the covariance matrix.

C. Cooperative strategies

We consider now a cooperative strategy where each node k has access to the information of its neighborhood $\mathcal{V}_k$, which is the subset of nodes directly connected to node k, including itself. The optimization problem can be presented as a minimization of the cost function $\sum_{k=1}^{N} J_k(\boldsymbol{w})$, where $J_k(\boldsymbol{w})$ is the cost function at node k, for instance (1). Since each node k communicates only with its neighboring nodes, we introduce the nonnegative coefficients c_{kl} that relate node k to its neighbors. They satisfy the following conditions:

$$c_{kl} \geq 0, \qquad \sum_{l \in \mathcal{V}_k} c_{kl} = 1, \qquad \text{and} \qquad c_{kl} = 0 \text{ if } l \notin \mathcal{V}_k. \tag{4}$$

It is worth noting that if $l \notin \mathcal{V}_k$, $k \notin \mathcal{V}_l$ and thus c_{lk} is also null. Using these coefficients, we define, for each node l, a local cost that consists of a weighted combination of the individual costs of the neighbors of node l (including l itself):

$$J_l^{\text{loc}}(\boldsymbol{w}) = \sum_{k \in \mathcal{V}_l} c_{kl}\, J_k(\boldsymbol{w}) = \sum_{k=1}^{N} c_{kl}\, J_k(\boldsymbol{w}). \tag{5}$$

It turns out that the cumulative sum of these local cost functions is equal to the global cost function, since we have

$$\sum_{k=1}^{N} J_k(\boldsymbol{w}) = \sum_{k=1}^{N} \Big(\sum_{l=1}^{N} c_{kl} \Big) J_k(\boldsymbol{w}) = \sum_{l=1}^{N} \sum_{k=1}^{N} c_{kl} J_k(\boldsymbol{w}) = \sum_{l=1}^{N} J_l^{\text{loc}}(\boldsymbol{w}).$$

For node k, the global cost function can be decomposed as

$$\sum_{l=1}^{N} J_l(\boldsymbol{w}) = J_k^{\text{loc}}(\boldsymbol{w}) + \sum_{\substack{l=1 \\ l \neq k}}^{N} J_l^{\text{loc}}(\boldsymbol{w}). \tag{6}$$

While the first term in the right-hand-side is known at the node under scrutiny, the second one should be estimated using information from the neighbors of node k; thus, the above summation is restricted to its neighborhood. Moreover, we relax it by constraining the norm between the estimated vector $\boldsymbol{w}$ and the optimal (global) principal axis $\boldsymbol{w}_*$ within the neighborhood of node k, with the following regularization:

$$\sum_{\substack{l=1 \\ l \neq k}}^{N} J_l^{\text{loc}}(\boldsymbol{w}) \approx \sum_{l \in \mathcal{V}_k \setminus \{k\}} b_{lk}\, \|\boldsymbol{w} - \boldsymbol{w}_*\|^2,$$

where the parameters b_{lk} control the tradeoff between the accuracy and the smoothness of the solution. Such regularization is also motivated by investigating the second-order Taylor expansion of $J_l^{\text{loc}}(\boldsymbol{w})$, as described in [21]. Note that we use $\boldsymbol{w}_*$ knowing that we do not have access to its value. We shall show in the following how to overcome this problem.

By injecting this approximation into (6), and using (5), the minimization of the global cost function at k is equivalent to minimizing

$$\begin{aligned} J_k^{\text{glob}}(\boldsymbol{w}) &= J_k^{\text{loc}}(\boldsymbol{w}) + \sum_{l \in \mathcal{V}_k \setminus \{k\}} b_{lk} \|\boldsymbol{w} - \boldsymbol{w}_*\|^2 \\ &= \sum_{l \in \mathcal{V}_k} c_{lk} J_l(\boldsymbol{w}) + \sum_{l \in \mathcal{V}_k \setminus \{k\}} b_{lk} \|\boldsymbol{w} - \boldsymbol{w}_*\|^2, \end{aligned} \tag{7}$$

where the last equality follows from (5). In order to minimize the above cost function, the node k applies the gradient descent to $J_k^{\text{glob}}(\boldsymbol{w})$ with:

$$\boldsymbol{w}_{k,t} = \boldsymbol{w}_{k,t-1} - \eta_{k,t}\, \nabla_{\boldsymbol{w}} J_k^{\text{glob}}(\boldsymbol{w}_{k,t-1}). \tag{8}$$

Here, $\eta_{k,t}$ is the learning rate for node k at iteration t. Replacing $J_k^{\text{glob}}(\boldsymbol{w})$ by its expression in (7), we get:

$$\boldsymbol{w}_{k,t} = \boldsymbol{w}_{k,t-1} - \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \nabla_{\boldsymbol{w}} J_l(\boldsymbol{w}_{t-1}) + \eta_{k,t} \sum_{l \in \mathcal{V}_k \setminus \{k\}} b_{lk} (\boldsymbol{w}_* - \boldsymbol{w}_{k,t-1}).$$

In this expression, $\nabla_{\boldsymbol{w}} J_l(\boldsymbol{w}_{t-1})$ is the gradient of the PCA-based cost function, presented earlier in (2). In this case, we get

$$\boldsymbol{w}_{k,t} = \boldsymbol{w}_{k,t-1} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \big(\boldsymbol{x}_l \, y_{\boldsymbol{w}_{t-1},l} - y^2_{\boldsymbol{w}_{t-1},l} \, \boldsymbol{w}_{l,t-1} \big) + \eta_{k,t} \sum_{l \in \mathcal{V}_k \backslash \{k\}} b_{lk} (\boldsymbol{w}_* - \boldsymbol{w}_{k,t-1}). \tag{9}$$

Without loss of generality, we shall consider the update rule (9) in the following. This update rule from $\boldsymbol{w}_{k,t-1}$ to $\boldsymbol{w}_{k,t}$ involves adding two correction terms: the first one operates in the maximum variance direction and the second one associates neighborhood regularization. By decomposing this calculation into two successive steps, we get two possible strategies, depending on the order of adding the correction terms, and that differ essentially in the approximation of the unknown $\boldsymbol{w}_*$ in (9), as shown next.

Adapt-then-combine (ATC) strategy

We express the update rule (9) as follows:

$$\begin{aligned}
\boldsymbol{\phi}_{k,t} &= \boldsymbol{w}_{k,t-1} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \Big(\boldsymbol{x}_l y_{\boldsymbol{w}_{t-1},l} - y^2_{\boldsymbol{w}_{t-1},l} \boldsymbol{w}_{l,t-1} \Big), \\
\boldsymbol{w}_{k,t} &= \boldsymbol{\phi}_{k,t} + \eta_{k,t} \sum_{l \in \mathcal{V}_k \backslash \{k\}} b_{lk} (\boldsymbol{w}_* - \boldsymbol{w}_{k,t-1}).
\end{aligned}$$

The first step uses local gradient vectors from the neighborhood of the node k in order to update $\boldsymbol{w}_{k,t-1}$ to the intermediate estimate $\boldsymbol{\phi}_{k,t}$. The second step updates $\boldsymbol{\phi}_{k,t}$ to $\boldsymbol{w}_{k,t}$. This second step is not realizable since the nodes do not know $\boldsymbol{w}_*$. However, each node l has its own approximation of $\boldsymbol{w}_*$, which is its local intermediate estimate $\boldsymbol{\phi}_{l,t}$. Therefore, we replace $\boldsymbol{w}_*$ by $\boldsymbol{\phi}_{l,t}$. On the other hand, the intermediate value $\boldsymbol{\phi}_{k,t}$ at node k is obtained by incorporating information from the neighbors (as given by the first step). Thus, it is generally a better estimate for $\boldsymbol{w}_*$ than $\boldsymbol{w}_{k,t-1}$. Therefore, we further replace $\boldsymbol{w}_{k,t-1}$ by $\boldsymbol{\phi}_{k,t}$ in the second step. Hence, the second step is rewritten as:

$$\begin{aligned}
\boldsymbol{w}_{k,t} &= \boldsymbol{\phi}_{k,t} + \eta_{k,t} \sum_{l \in \mathcal{V}_k \backslash \{k\}} b_{lk} \, (\boldsymbol{\phi}_{l,t} - \boldsymbol{\phi}_{k,t}) \\
&= \Big(1 - \eta_{k,t} \sum_{l \in \mathcal{V}_k \backslash \{k\}} b_{lk} \Big) \, \boldsymbol{\phi}_{k,t} + \eta_{k,t} \sum_{l \in \mathcal{V}_k \backslash \{k\}} b_{lk} \, \phi_{l,t},
\end{aligned}$$

or, equivalently: $\boldsymbol{w}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl} \, \boldsymbol{\phi}_{l,t}$ where we have used the following nonnegative weighting coefficients:

$$a_{kl} = \begin{cases} 1 - \eta_{k,t} \sum_{i \in \mathcal{V}_k \backslash \{k\}} b_{ik}, & \text{if } l = k; \\ \eta_{k,t} \, b_{lk}, & \text{if } l \in \mathcal{V}_k \backslash \{k\} \ ; \\ 0, & \text{otherwise.} \end{cases}$$

Finally, the update rule for the adapt-then-combine strategy is:

$$\begin{aligned}
\boldsymbol{\phi}_{k,t} &= \boldsymbol{w}_{k,t-1} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \Big(\boldsymbol{x}_l y_{\boldsymbol{w}_{t-1},l} - y^2_{\boldsymbol{w}_{t-1},l} \boldsymbol{w}_{l,t-1} \Big), \\
\boldsymbol{w}_{k,t} &= \sum_{l \in \mathcal{V}_k} a_{kl} \, \boldsymbol{\phi}_{l,t}.
\end{aligned}$$

In the special case where no information exchange is performed between nodes, except for the aggregation step, *i.e.*, $c_{kk} = 1$ and $c_{kl} = 0$ for all $l \neq k$, we get:

$$\begin{aligned}
\boldsymbol{\phi}_{k,t} &= \boldsymbol{w}_{k,t-1} + \eta_{k,t} \left(\boldsymbol{x}_k \, y_{\boldsymbol{w}_{t-1},k} - y^2_{\boldsymbol{w}_{t-1},k} \, \boldsymbol{w}_{k,t-1} \right) \\
\boldsymbol{w}_{k,t} &= \sum_{l \in \mathcal{V}_k} a_{kl} \, \boldsymbol{\phi}_{l,t}.
\end{aligned}$$

Note that such particular configuration would allow minimizing the amount of information exchanged between nodes, by excluding the transmission of the estimates y and the measurements $\boldsymbol{x}$ within the network, and restricting the exchanges to $\boldsymbol{w}$ (or equivalently $\boldsymbol{\phi}$).

Combine-then-adapt (CTA) strategy

In this strategy, we express the update rule (9) as follows:

$$\boldsymbol{\phi}_{k,t} = \boldsymbol{w}_{k,t-1} + \eta_{k,t} \sum_{l \in \mathcal{V}_k \setminus \{k\}} b_{lk}(\boldsymbol{w}_* - \boldsymbol{w}_{k,t-1})$$
$$\boldsymbol{w}_{k,t} = \boldsymbol{\phi}_{k,t} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \Big(\boldsymbol{x}_l \, y_{\boldsymbol{w}_{t-1},l} - y^2_{\boldsymbol{w}_{t-1},l} \, \boldsymbol{w}_{l,t-1} \Big).$$

For the same reasons shown in the adapt-then-combine strategy, we replace in the first step $\boldsymbol{w}_*$ by $\boldsymbol{w}_{l,t-1}$ and, in the second step, we replace $\boldsymbol{w}_{l,t-1}$ by $\boldsymbol{\phi}_{l,t}$. Introducing the same coefficients a_{kl}, we obtain the update rule for the combine-then-adapt strategy:

$$\boldsymbol{\phi}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl} \, \boldsymbol{w}_{l,t-1},$$
$$\boldsymbol{w}_{k,t} = \boldsymbol{\phi}_{k,t} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \Big(\boldsymbol{x}_l \, y_{\boldsymbol{w}_{t-1},l} - y^2_{\boldsymbol{w}_{t-1},l} \, \boldsymbol{\phi}_{l,t} \Big).$$

In the case of no information exchange except for the aggregation step, we obtain:

$$\boldsymbol{\phi}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl} \, \boldsymbol{w}_{l,t-1}$$
$$\boldsymbol{w}_{k,t} = \boldsymbol{\phi}_{k,t} + \eta_{k,t} \Big(\boldsymbol{x}_k \, y_{\boldsymbol{w}_{t-1},k} - y^2_{\boldsymbol{w}_{t-1},k} \, \boldsymbol{\phi}_{k,t} \Big).$$

D. Connections to the consensus strategies

The consensus-type implementation [22] is a class of distributed strategies that reveals the coordination of networks of autonomous agents. In order to estimate a common parameter $\boldsymbol{w}_*$, the consensus allows each node k to align its decision $\boldsymbol{w}_{k,t}$ with the decisions of its neighbors:

$$\boldsymbol{w}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl} \boldsymbol{w}_{k,t-1},$$

where a_{kl} are nonnegative weights that node k gives to the estimates received from its neighbors. In order to distribute the computations among the nodes, the authors in [23], [24], [25] propose to combine the consensus and the gradient descent approaches. In such algorithm, each node k applies the negative gradient step to the average of its neighbors' estimates. Therefore, the node k aligns its decision $\boldsymbol{w}_{k,t}$ with the decisions of its neighbors and minimizes its own objective $J_k(\boldsymbol{w})$ using:

$$\boldsymbol{w}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl} \boldsymbol{w}_{k,t-1} - \eta_{k,t} \, \nabla_{\boldsymbol{w}} J_k(\boldsymbol{w}_{k,t-1}).$$

In this context, we revisit the estimation of the principal axis with the proposed cost function given in (1). This strategy takes the following form:

$$\boldsymbol{w}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl} \boldsymbol{w}_{k,t-1} + \eta_{k,t} \Big(\boldsymbol{x}_k \, y_{\boldsymbol{w}_{t-1},k} - \, y^2_{\boldsymbol{w}_{t-1},k} \boldsymbol{w}_{k,t-1} \Big).$$

The consensus strategy, the adapt-then-combine, and the combine-then-adapt diffusion strategies have the same computational complexity. However, the diffusion strategies outperform the consensus implementation. Indeed, the adapt-then-combine and the combine-then-adapt diffusion strategies can be expressed respectively by:

$$\boldsymbol{w}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl} \Big(\boldsymbol{w}_{k,t-1} + \eta_{k,t} \left(\boldsymbol{x}_k \, y_{\boldsymbol{w},k} - \, y^2_{\boldsymbol{w},k} \boldsymbol{w}_{k,t-1} \right) \Big),$$
$$\boldsymbol{w}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl} \boldsymbol{w}_{k,t-1} + \eta_{k,t} \Big(\boldsymbol{x}_k \, y_{\boldsymbol{w},k} - \, y^2_{\boldsymbol{w},k} \sum_{l \in \mathcal{V}_k} a_{kl} \boldsymbol{w}_{k,t-1} \Big).$$

Considering for instance the combine-then-adapt strategy, we note that the combination coefficients a_{kl} appear in the correction term, while the consensus uses only $\boldsymbol{w}_{k,t-1}$ to correct the error (*i.e.*, without involving the neighborhood), which is less effective, as shown in the experimental results.

IV. Other cost functions

In this part, we extend the study of Section III to include different cost functions. These cost functions are chosen in the light of some known rules, as described next.

A. Information theory (IT)

Drawing inspiration from the information theoretical framework studied in [16], [26], we consider the maximization of the entropy, defined by $\ln(y_{\boldsymbol{w},k}^2)$, under a constraint on the norm of $\boldsymbol{w}$. To this end, we minimize the following cost function associated with node k:

$$J_k(\boldsymbol{w}) = \tfrac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \tfrac{1}{2}\ln(y_{\boldsymbol{w},k}^2), \tag{10}$$

whose gradient with respect to $\boldsymbol{w}$ is

$$\nabla_{\boldsymbol{w}} J_k(\boldsymbol{w}) = \boldsymbol{w} - \frac{\boldsymbol{x}_k}{y_{\boldsymbol{w},k}}. \tag{11}$$

By applying the gradient descent technique to this cost function, we obtain the following noncooperative update rule:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \eta_t \Big(\frac{\boldsymbol{x}_k}{y_{\boldsymbol{w}_{t-1},k}} - \boldsymbol{w}_{t-1}\Big).$$

Cooperative strategies are also investigated by information diffusion. The adapt-then-combine update rule becomes in this case:

$$\begin{aligned}
\boldsymbol{\phi}_{k,t} &= \boldsymbol{w}_{k,t-1} + \eta_{k,t} \sum_{l\in\mathcal{V}_k} c_{lk}\Big(\frac{\boldsymbol{x}_l}{y_{\boldsymbol{w}_{t-1},l}} - \boldsymbol{w}_{l,t-1}\Big),\\
\boldsymbol{w}_{k,t} &= \sum_{l\in\mathcal{V}_k} a_{kl}\,\boldsymbol{\phi}_{l,t}.
\end{aligned}$$

The combine-then-adapt update rule becomes:

$$\begin{aligned}
\boldsymbol{\phi}_{k,t} &= \sum_{l\in\mathcal{V}_k} a_{kl}\,\boldsymbol{w}_{l,t-1},\\
\boldsymbol{w}_{k,t} &= \boldsymbol{\phi}_{k,t} + \eta_{k,t} \sum_{l\in\mathcal{V}_k} c_{lk}\Big(\frac{\boldsymbol{x}_l}{y_{\boldsymbol{w}_{t-1},l}} - \boldsymbol{w}_{l,t-1}\Big).
\end{aligned}$$

While the cost function (1) studied in Section III and the cost function (12) inspired from the information theory are very different, the corresponding update rules are intimately connected. Indeed, the latter can be derived from the former by using the normalized learning rate $\eta_t/y_{\boldsymbol{w}_{t-1},k}^2$.

B. Rayleigh quotient (RQ)

The eigen-decomposition problem is highly connected to the Rayleigh quotient. Inspired from the Rayleigh quotient framework studied in [17], [18], we minimize the following cost function associated with node k:

$$J_k(\boldsymbol{w}) = -\frac{y_{\boldsymbol{w},k}^2}{2\,\boldsymbol{w}^\top \boldsymbol{w}}, \tag{12}$$

whose gradient with respect to $\boldsymbol{w}$ is

$$\nabla_{\boldsymbol{w}} J_k(\boldsymbol{w}) = \frac{1}{\boldsymbol{w}^\top \boldsymbol{w}}\Big(\boldsymbol{w}\frac{y_{\boldsymbol{w},k}^2}{\boldsymbol{w}^\top \boldsymbol{w}} - \boldsymbol{x}_k y_{\boldsymbol{w},k}\Big). \tag{13}$$

The gradient descent technique, applied to this cost function, leads to the following noncooperative update rule:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \eta_t \frac{1}{\boldsymbol{w}_{t-1}^\top \boldsymbol{w}_{t-1}}\Big(\boldsymbol{x}_k y_{\boldsymbol{w}_{t-1},k} - \boldsymbol{w}_{t-1}\frac{y_{\boldsymbol{w}_{t-1},k}^2}{\boldsymbol{w}_{t-1}^\top \boldsymbol{w}_{t-1}}\Big).$$

By including a diffusion process, we propose two cooperative strategies. In the case of the adapt-then-combine update rule, we get:

$$\boldsymbol{\phi}_{k,t} = \boldsymbol{w}_{k,t-1} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \frac{1}{\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}} \Big(\boldsymbol{x}_l y_{\boldsymbol{w}_{t-1},l} - \boldsymbol{w}_{l,t-1} \frac{y^2_{\boldsymbol{w}_{t-1},l}}{\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}} \Big),$$

$$\boldsymbol{w}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl}\, \boldsymbol{\phi}_{l,t}.$$

For the combine-then-adapt update rule, we have:

$$\boldsymbol{\phi}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl}\, \boldsymbol{w}_{l,t-1},$$

$$\boldsymbol{w}_{k,t} = \boldsymbol{\phi}_{k,t} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \frac{1}{\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}} \Big(\boldsymbol{x}_l y_{\boldsymbol{w}_{t-1},l} - \boldsymbol{w}_{l,t-1} \frac{y^2_{\boldsymbol{w}_{t-1},l}}{\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}} \Big).$$

C. OJAn and LUO algorithms

The OJAn and LUO algorithms, presented by Luo *et al.* in [18], are inter-connected to the Rayleigh quotient [17], [27]. The OJAn algorithm is a normalized version of Oja's rule. It can be obtained from the Rayleigh quotient by using a learning rate $\eta_t(\boldsymbol{w}_{t-1}^\top \boldsymbol{w}_{t-1})$. By revisiting the gradient descent formulation (3) in the light of this learning rate, we get the following noncooperative update rule:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \eta_t \Big(\boldsymbol{x}_k y_{\boldsymbol{w}_{t-1},k} - \boldsymbol{w}_{t-1} \frac{y^2_{\boldsymbol{w}_{t-1},k}}{\boldsymbol{w}_{t-1}^\top \boldsymbol{w}_{t-1}} \Big).$$

In this case, the adapt-then-combine update rule becomes:

$$\boldsymbol{\phi}_{k,t} = \boldsymbol{w}_{k,t-1} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \Big(\boldsymbol{x}_l y_{\boldsymbol{w}_{t-1},l} - \boldsymbol{w}_{l,t-1} \frac{y^2_{\boldsymbol{w}_{t-1},l}}{\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}} \Big),$$

$$\boldsymbol{w}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl}\, \boldsymbol{\phi}_{l,t}.$$

The combine-then-adapt update rule becomes:

$$\boldsymbol{\phi}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl}\, \boldsymbol{w}_{l,t-1},$$

$$\boldsymbol{w}_{k,t} = \boldsymbol{\phi}_{k,t} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} \Big(\boldsymbol{x}_l y_{\boldsymbol{w}_{t-1},l} - \boldsymbol{w}_{l,t-1} \frac{y^2_{\boldsymbol{w}_{t-1},l}}{\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}} \Big).$$

The LUO algorithm can be derived from the Rayleigh quotient by using the learning rate $\eta_t(\boldsymbol{w}_{t-1}^\top \boldsymbol{w}_{t-1})^2$. This leads to following noncooperative update rule:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \eta_t\, (\boldsymbol{w}_{t-1}^\top \boldsymbol{w}_{t-1}) \Big(\boldsymbol{x}_k y_{\boldsymbol{w}_{t-1},k} - \boldsymbol{w}_{t-1} \frac{y^2_{\boldsymbol{w}_{t-1},k}}{\boldsymbol{w}_{t-1}^\top \boldsymbol{w}_{t-1}} \Big),$$

as well as the two cooperative strategies, the adapt-then-combine

$$\boldsymbol{\phi}_{k,t} = \boldsymbol{w}_{k,t-1} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} (\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}) \Big(\boldsymbol{x}_l y_{\boldsymbol{w}_{t-1},l} - \boldsymbol{w}_{l,t-1} \frac{y^2_{\boldsymbol{w}_{t-1},l}}{\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}} \Big),$$

$$\boldsymbol{w}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl}\, \boldsymbol{\phi}_{l,t}.$$

and the combine-then-adapt update rule

$$\boldsymbol{\phi}_{k,t} = \sum_{l \in \mathcal{V}_k} a_{kl}\, \boldsymbol{w}_{l,t-1},$$

$$\boldsymbol{w}_{k,t} = \boldsymbol{\phi}_{k,t} + \eta_{k,t} \sum_{l \in \mathcal{V}_k} c_{lk} (\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}) \Big(\boldsymbol{x}_l y_{\boldsymbol{w}_{t-1},l} - \boldsymbol{w}_{l,t-1} \frac{y^2_{\boldsymbol{w}_{t-1},l}}{\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{l,t-1}} \Big).$$

V. Extracting multiple principal axes

This section extends the previous derivations in Section III to multiple principal axes, generalization to other cost functions as given in Section IV is straightforward. To this end, we denote by

$$\boldsymbol{W}_t = [\boldsymbol{w}_{1,t} \quad \boldsymbol{w}_{2,t} \quad \cdots \quad \boldsymbol{w}_{r,t}]^\top$$

the r-by-p matrix of the first r principal axes estimated at iteration t, sorted in descending order of their eigenvalues. Let

$$\boldsymbol{y}_{\boldsymbol{W},k} = [y_{\boldsymbol{w}_1,k} \quad y_{\boldsymbol{w}_2,k} \quad \cdots \quad y_{\boldsymbol{w}_r,k}]^\top,$$

where $y_{\boldsymbol{w}_j,k} = \boldsymbol{w}_{j,t}^\top \boldsymbol{x}_k$. Next, we combine the update rules given in Sections III-B and III-C with several orthogonalization processes, including the well-known Gram-Schmidt orthogonalization and the deflation [28]. Note that other orthogonalization processes exist in the literature, such as the symmetric orthogonalization [29]. However, the latter is inappropriate in network applications due to its high computational complexity.

A. Gram-Schmidt orthogonalization process

We use a generalized Hebbian approach as proposed by Sanger for the linear principal component analysis [30], [31]. In the noncooperative strategy, we write the update rule of the j-th principal axis as follows:

$$\boldsymbol{w}_{j,t} = \boldsymbol{w}_{j,t-1} + \eta_t \left(\boldsymbol{x}_k \, y_{\boldsymbol{w}_j,k} - y_{\boldsymbol{w}_j,k} \sum_{l=1}^{j} y_{\boldsymbol{w}_j,l} \, \boldsymbol{w}_{l,t-1} \right). \tag{14}$$

It is clear how the estimation of the j-th component $\boldsymbol{w}_{j,t}$ at iteration t involves the previous estimates (at iteration $t-1$) of lower-order components, *i.e.*, $\boldsymbol{w}_{l,t-1}$, for $l = 1, \ldots, j$. By collecting all these estimates in a single matrix $\boldsymbol{W}_t$, we obtain the following update rule:

$$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} + \eta_t \left(\boldsymbol{y}_{\boldsymbol{W}_{t-1},k} \, \boldsymbol{x}_k^\top - \mathrm{LT}(\boldsymbol{y}_{\boldsymbol{W}_{t-1},k} \, \boldsymbol{y}_{\boldsymbol{W}_{t-1},k}^\top) \, \boldsymbol{W}_{t-1} \right), \tag{15}$$

where $\mathrm{LT}(\cdot)$ makes its argument lower triangular by setting to zero the entries above its diagonal. Note that the learning rate does not need to be the same for all the principal axes.

In the cooperative strategies, we denote by $\boldsymbol{\Phi}_{k,t} = [\boldsymbol{\phi}_{1,k,t} \quad \boldsymbol{\phi}_{2,k,t} \quad \cdots \quad \boldsymbol{\phi}_{r,k,t}]^\top$ the r-by-p matrix of the r intermediate estimates. In matrix form, the update rule associated with the adapt-then-combine strategy is as follows

$$\begin{aligned}
\boldsymbol{\Phi}_{k,t} &= \boldsymbol{W}_{k,t-1} + \eta_{k,t} \Big(\boldsymbol{y}_{\boldsymbol{W}_{t-1},k} \boldsymbol{x}_k^\top - \mathrm{LT}(\boldsymbol{y}_{\boldsymbol{W}_{t-1},k} \boldsymbol{y}_{\boldsymbol{W}_{t-1},k}^\top) \, \boldsymbol{W}_{k,t-1} \Big), \\
\boldsymbol{W}_{k,t} &= \sum_{l \in \mathcal{V}_k} a_{kl} \, \boldsymbol{\Phi}_{l,t}.
\end{aligned}$$

The update rule for the combine-then-adapt strategy becomes

$$\begin{aligned}
\boldsymbol{\Phi}_{k,t} &= \sum_{l \in \mathcal{V}_k} a_{kl} \, \boldsymbol{W}_{l,t-1}, \\
\boldsymbol{W}_{k,kt} &= \boldsymbol{\Phi}_{k,t} + \eta_{k,t} \Big(\boldsymbol{y}_{\boldsymbol{W}_{t-1},k} \boldsymbol{x}_k^\top - \mathrm{LT}(\boldsymbol{y}_{\boldsymbol{W}_{t-1},k} \boldsymbol{y}_{\boldsymbol{W}_{t-1},k}^\top) \, \boldsymbol{\Phi}_{k,t} \Big).
\end{aligned}$$

B. Deflation scheme

Like the Gram-Schmidt orthogonalization, the deflation scheme updates all estimates at each iteration. However, it proceeds in two consecutive steps, within each iteration. First, intermediate estimates $\boldsymbol{w}_{j,\star}$, for $j = 1, \ldots, r$ are independently calculated using:

$$\boldsymbol{w}_{j,\star} = \boldsymbol{w}_{j,t-1} + \eta_t \left(\boldsymbol{x}_k \, y_{\boldsymbol{w}_j,k} - y_{\boldsymbol{w}_j,k}^2 \, \boldsymbol{w}_{j,t-1} \right). \tag{16}$$

Let $\boldsymbol{W}_\star = [\boldsymbol{w}_{1,\star} \quad \boldsymbol{w}_{2,\star} \quad \cdots \quad \boldsymbol{w}_{r,\star}]$ be the matrix of the intermediate estimates. We can write:

$$\boldsymbol{W}_\star = \boldsymbol{W}_{t-1} + \eta_t \Big(\boldsymbol{x}_k \boldsymbol{y}_{\boldsymbol{W},k}^\top - \boldsymbol{W}_{t-1} \mathrm{diag}(\boldsymbol{y}_{\boldsymbol{W},k} \, \boldsymbol{y}_{\boldsymbol{W},k}^\top) \Big),$$

where $\mathrm{diag}(\cdot)$ makes its argument diagonal by setting all non-diagonal entries to zero.

TABLE I
EXPRESSIONS OF THE TIME CONSTANTS FOR THE FIRST AND THE i-TH PRINCIPAL AXES, FOR EACH OF THE ALGORITHMS STUDIED IN THIS PAPER.

Algorithm	Time constant for $\boldsymbol{w}_1$	Time constant for $\boldsymbol{w}_i$
Oja	$1/\lambda_1$	$1/(\lambda_1 - \lambda_i)$
IT	1	1
RQ	$\|\boldsymbol{w}_{k,0}\|^2/\lambda_1$	$\|\boldsymbol{w}_{k,0}\|^2/(\lambda_1 - \lambda_i)$
OJAn	$1/\lambda_1$	$1/(\lambda_1 - \lambda_i)$
LUO	$\|\boldsymbol{w}_{k,0}\|^{-2}/\lambda_1$	$\|\boldsymbol{w}_{k,0}\|^{-2}/(\lambda_1 - \lambda_i)$

Then, each estimate is updated by the node k using the intermediate estimates from (16) as follows:

$$\boldsymbol{w}_{j,t} = \boldsymbol{w}_{j,\star} - \sum_{l=1}^{j-1} (\boldsymbol{w}_{l,t-1}^\top \boldsymbol{w}_{j,\star})\, \boldsymbol{w}_{l,t-1}.$$

By regrouping all these vectors $\boldsymbol{w}_{j,t}$ in a matrix $\boldsymbol{W}_t$, this expression for noncooperative strategy is written in matrix form as:

$$\boldsymbol{W}_t = \boldsymbol{W}_\star - \boldsymbol{W}_{t-1}\Big(\boldsymbol{W}_{t-1}^\top \boldsymbol{W}_\star - \mathrm{LT}(\boldsymbol{W}_{t-1}^\top \boldsymbol{W}_\star)\Big).$$

A normalization of $\boldsymbol{w}_{j,t}$ is required by the orthogonalization procedure, by operating the projection $\frac{\boldsymbol{w}_{j,t}}{\|\boldsymbol{w}_{j,t}\|}$.

Note that this scheme could be applied in a sequential way: The node k can extract the principal axes one after the other, by operating on a simple vector update. For the j-th principal axis, for $j = 1, \ldots, r$, the intermediate j-th estimate defined in (16) is used. Then, the node under scrutiny updates the j-th principal axis with the following update rule:

$$\boldsymbol{w}_{j,t} = \boldsymbol{w}_\star - \sum_{l=1}^{j-1} (\boldsymbol{w}_{l,\infty}^\top \boldsymbol{w}_{j,\star})\, \boldsymbol{w}_{l,\infty}, \tag{17}$$

where $\boldsymbol{w}_{l,\infty}$ is the extracted l-th principal axis obtained after convergence of $\boldsymbol{w}_{l,t}$, for $l = 1, \ldots, j-1$. However, such process is ill-suited for real-time network applications, because the extraction of the $(j+1)$-th principal axis cannot start before the convergence of $\boldsymbol{w}_{j,t}$.

For the cooperative strategies, the deflation scheme can be easily incorporated in the adapt-then-combine strategy, with

$$\left\{\begin{array}{rl} \boldsymbol{\Phi}_{k,\star} & = \boldsymbol{W}_{k,t-1} + \eta_{k,t}\Big(\boldsymbol{x}_k \boldsymbol{y}_{\boldsymbol{W}_{t-1},k}^\top - \boldsymbol{W}_{k,t-1}\mathrm{diag}(\boldsymbol{y}_{\boldsymbol{W}_{t-1},k}\, \boldsymbol{y}_{\boldsymbol{W}_{t-1},k}^\top)\Big) \\ \boldsymbol{\Phi}_{k,t} & = \boldsymbol{\Phi}_\star - \boldsymbol{\Phi}_{k,t-1}\Big(\boldsymbol{\Phi}_{k,t-1}^\top \boldsymbol{\Phi}_{k,\star} - \mathrm{LT}(\boldsymbol{\Phi}_{k,t-1}^\top \boldsymbol{\Phi}_{k,\star})\Big) \\ \boldsymbol{W}_{k,t} & = \displaystyle\sum_{l\in\mathcal{V}_k} a_{kl}\, \boldsymbol{\Phi}_{l,t}. \end{array}\right.$$

as well as the combine-then-adapt strategy, with

$$\begin{array}{rl} \boldsymbol{\Phi}_{k,t-1} & = \displaystyle\sum_{l\in\mathcal{V}_k} a_{kl}\, \boldsymbol{W}_{l,t-1} \\ \left\{\begin{array}{rl} \boldsymbol{W}_{k,\star} & = \boldsymbol{\Phi}_{k,t-1} + \eta_{k,t}\Big(\boldsymbol{x}_k \boldsymbol{y}_{\boldsymbol{W}_{t-1},k}^\top - \boldsymbol{\Phi}_{k,t-1}\mathrm{diag}(\boldsymbol{y}_{\boldsymbol{W}_{t-1},k}\, \boldsymbol{y}_{\boldsymbol{W}_{t-1},k}^\top)\Big) \\ \boldsymbol{W}_{k,t} & = \boldsymbol{W}_\star - \boldsymbol{W}_{k,t-1}\Big(\boldsymbol{W}_{k,t-1}^\top \boldsymbol{W}_{k,\star} - \mathrm{LT}(\boldsymbol{W}_{k,t-1}^\top \boldsymbol{W}_{k,\star})\Big). \end{array}\right. \end{array}$$

Remarks on the convergence

The previous study takes into account the cost function (1) and can be easily generalized to other algorithms by considering the cost function. In the light of the work in [32], it is worth noting some observations on the convergence of these algorithms. We refer the reader to [32] and references therein for more details.

The convergence of the IT algorithm is independent of the largest eigenvalue λ_1. The convergence of all other algorithms increases with λ_1 and λ_1/λ_2, when extracting the first and second principal axes. Moreover, the convergence of Oja's based rule and the OJAn algorithm cannot be improved by increasing $\|\boldsymbol{w}_{k,0}\|$, while the convergence of the LUO algorithm decreases for smaller $\|\boldsymbol{w}_{k,0}\|$ and the convergence of the RQ algorithm decreases for larger $\|\boldsymbol{w}_{k,0}\|$. TABLE I shows the the rate of convergence in terms of the expressions of the time constants for the first and the i-th principal axes, for all the algorithms studied in this paper.

The algorithms associated to Oja's rule and to the IT converge to unit norm vectors, *i.e.*, $\|\boldsymbol{w}_{k,\infty}\| \to 1$. On the other hand, the RQ, OJAn, and LUO algorithms require a normalization $\|\boldsymbol{w}_{k,0}\| = 1$ for the principal eigenvector $\boldsymbol{w}_{k,\infty}$ to converge to a unit-norm solution.

VI. Comparison with other algorithms

In this section, we present two algorithms well-known in the literature, *collective-PCA* and *PCA-based distributed approach*. As opposed to these algorithms where one or many eigen-decomposition problems are solved, the in-network noncooperative and cooperative strategies proposed in this paper neither estimate the covariance matrix nor operate an eigen-decomposition. Experimental results are given in the following section.

A. Collective-PCA

The collective-PCA (CPCA) introduced in [33] essentially operates in two stages: pre-processing the data in cluster-heads (*i.e.*, local fusion centers) before sending them to a FC. Let $\boldsymbol{X}$ be the N-by-p matrix of the entire data, *i.e.*, $\boldsymbol{X} = [\boldsymbol{x}_1 \;\; \boldsymbol{x}_2 \;\cdots\; \boldsymbol{x}_N]^\top$. This matrix is divided into d submatrix: $\boldsymbol{X} = [\boldsymbol{X}_1 \;\; \boldsymbol{X}_2 \;\cdots\; \boldsymbol{X}_d]$, where $\boldsymbol{X}_i$ is a N-by-p_i submatrix of $\boldsymbol{X}$, and $\sum_{i=1}^{d} p_i = p$. A local centralized PCA is performed on the data of each $\boldsymbol{X}_i$. Let $\boldsymbol{W}^i$ be the p_i-by-r_i matrix of the r_i first principal axes, and $\boldsymbol{W}^{tot}$ be a diagonal matix with matrices $\boldsymbol{W}^1, \ldots, \boldsymbol{W}^d$ on the diagonal. The principal component corresponding to $\boldsymbol{X}_i$, $\boldsymbol{Y} = \boldsymbol{X}_i\boldsymbol{W}^i$ is sent to the FC and collected into N-by-r matrix $\boldsymbol{Y} = [\boldsymbol{Y}_1 \;\; \boldsymbol{Y}_2 \;\cdots\; \boldsymbol{Y}_d]$, where $r = \sum_{i=1}^{d} r_i$. The FC performs a PCA on $\boldsymbol{Y}$ and extract its principal axes $\boldsymbol{v}_i$. Finally, the principal axes of $\boldsymbol{X}$ are $\boldsymbol{w}_i = \boldsymbol{W}^{tot}\boldsymbol{v}_i$. This centralized algorithm reduces the complexity of the communication with the FC. However, it still has an important computational complexity compared to our algorithms, because it requires to perform centralized PCA on different stages.

B. PCA-based distributed approach

The PCA-based distributed approach (PCADID) proposed in [34] is a distributed method. The N nodes are divided into d groups of n_i nodes, for $i = 1, \ldots, d$ where $\sum_{i=1}^{d} n_i = N$. Each group i deals p_i features where $\sum_{i=1}^{d} p_i = p$. Therefore, $\boldsymbol{X} = [\boldsymbol{X}_1 \;\; \boldsymbol{X}_2 \;\cdots\; \boldsymbol{X}_d]$, where $\boldsymbol{X}_i$ is a n_i-by-p_i submatrix of $\boldsymbol{X}$. First, for each group, data are normalized to the range $[0\,,1]$ and then one computes the column-centered matrix $\boldsymbol{X}_i^c$. For each group i, The principal axes of $\boldsymbol{X}_i^c$ are extracted using a singular value decomposition: $\boldsymbol{X}_i^c = \boldsymbol{U}_i\,\boldsymbol{\Sigma}_i\,\boldsymbol{V}_i^\top$. The matrix of the principal axes is given by the matrix $\boldsymbol{V}_i$. Each group sends this matrix to its neighbor group according to a routing process. Finally, the principal axes of $\boldsymbol{X}$ are $\boldsymbol{W} = \bigcup_i \boldsymbol{V}_i$. The PCADID operates through a "divide-to-conquer" scheme in order to reduce the computational complexity of the eigen-decomposition problem. However, it still requires the eigen-decomposition of several matrices, making it inappropriate for large scale networks. Moreover, our approach outperforms the PCADID technique as shown with experimental results in the following section; see for instance Fig. 2.

VII. Experimental results

In this section, the performance of the proposed approach is illustrated within two different contexts: dimensionality reduction of time series in wireless sensor networks and image processing. The performance is measured in terms of the angle between the principal axis $\boldsymbol{w}_*$, obtained from the centralized strategy with an eigen-decomposition of the covariance matrix, and the estimate $\boldsymbol{w}_{*,l}$ at node l, namely

$$\Theta_i = \arccos\left(\frac{\boldsymbol{w}_{*,l}^\top \boldsymbol{w}_*}{\|\boldsymbol{w}_{*,l}\|\|\boldsymbol{w}_*\|}\right). \tag{18}$$

In order to provide a fair comparative study, we use the same initial random estimate for all strategies.

A. Time series measurements in a WSN

In this section, we consider the problem of tracking a gas spread using a wireless sensor network (WSN) [35]. The sensor k at a position denoted by $\boldsymbol{z}_k \in \mathbb{Z}$ measures, at time θ, a gas quantity denoted by $x_{k,\theta}$. The region under scrutiny $\mathbb{Z} = [-0.5, 0.5] \times [-0.5, 0.5]$ is a two-dimensional unit-area. We aim to reduce the order of the time series of the measurements. The gas diffusion within this region is governed by the following differential equation:

$$\frac{\partial G(\boldsymbol{z},\theta)}{\partial \theta} - c\,\nabla^2_{\boldsymbol{z}} G(\boldsymbol{z},\theta) = Q(\boldsymbol{z},\theta),$$

where $G(\boldsymbol{z},\theta)$ is the density of gas depending on the position $\boldsymbol{z}$ and time θ, $\nabla^2_{\boldsymbol{z}}$ is the Laplace operator, c is the conductivity of the medium, and $Q(\boldsymbol{z},\theta)$ corresponds to the added quantity of gas. A gas source placed at the origin is activated from $\theta = 1$s to $\theta = 15$s. We use $N = 100$ sensors uniformly deployed in the region $\mathbb{Z}$, each acquiring a time series of 15 measurements, between $\theta = 1$s and $\theta = 15$s.

We consider a predetermined range of communication in the WSN: two nodes are considered as being connected when they are less than s units of distance apart, that is

$$\mathcal{V}_k = \{l \ : \ \|\boldsymbol{z}_k - \boldsymbol{z}_l\| < s\}.$$

In our experiments, we set this threshold to $s = 0.38$. On the other hand, throughout our simulations, we have noted that the value of the second eigenvector λ_2 is generally very small ($\lambda_2 \approx 0$). The reason behind this is that the rate of the measurement increase of each sensor, between two consecutive moments, is approximately the same for all sensors. This behavior is inherent to the nature of gas propagation in the space. Therefore, the second principal axis is difficult to be determined in this case. As for the choice of the stepsize parameters $\eta_{k,t}$, we use a cross validation approach: we vary the stepsize according to three series of values, the first being from $0.1, 0.01, \ldots, 0.1 \cdot 10^{-12}$, the second from $0.25, 0.025, \ldots, 0.25 \cdot 10^{-12}$, and the third from $0.5, 0.05, \ldots, 0.5 \cdot 10^{-12}$. Then, we choose the step size which leads to the best result in the noncooperative strategy. Note that the noncooperative strategy is used at this phase because of its smaller computational cost, compared to the other strategies. We consider a stepsize $\eta_1 = 0.0025$ for the first principal axis. The same stepsize value is taken for the cooperative strategies.For the information theory, we use $\eta_1 = 0.025$, for OJAn algorithm $\eta_1 = 0.005$, for LUO algorithm $\eta_1 = 0.0025$ for both cooperative and noncooperative strategies.

Fig. 2 and Fig. 3 show the convergence of each strategy for Oja's rule, the information theory, the Rayleigh quotient, the OJAn, and the LUO algorithms respectively, in terms of the angle Θ_1 for the first axis, averaged over the N nodes. The confronted strategies are the noncooperative strategy, the combine-then-adapt (CTA) and adapt-then-combine (ATC) diffusion strategies, and the consensus implementation. For the diffusion strategies, we take the simplified case where no information exchange is performed between nodes, except for the combination step; in other words, we use $c_{kk} = 1$, and $c_{lk} = 0$ for each $k \neq l$, with $l, k = 1, \ldots, 100$. Results are shown in terms of the angle averaged over all the nodes. Moreover, Fig. 2 shows that our strategies outperforms the PCADID. These learning curves show that the noncooperative strategy is outperformed by any diffusion strategy. The analysis of the diffusion strategies shows that the adapt-then-combine strategy performs slightly better that the combine-then-adapt strategy. The overall efficiency of the diffusion strategies is shown in terms of stability. On the other hand, considering the rate of convergence, the figures show that the LUO algorithm has the fastest convergence, followed consecutively by OJAn, Oja, RQ and IT algorithms. Oja and OJAn have the same behavior in terms of speed of convergence, RQ and LUO have a conflicting behavior as expected from TABLE I. Another

criterion for comparison is the level of the floor. The IT algorithm has the largest average angle at the convergence. All the other algorithms have comparable levels of the floor. Note that the consensus with RQ, OJAn, and LUO has a smaller average angle at the convergence than the other strategies, and it is slower in convergence than the other strategies with Oja, OJAn and LUO. Altogether, we can say that the best combinations that allow for a small angle floor, together with an acceptable rate of convergence, are the Oja and OJAn rules with the adapt-then-combine diffusion strategy.

B. Image processing application

In this section, we study an image processing application. For this purpose, we consider a dataset of the handwritten digit "1" given by images of 28-by-28 pixels, treated as 784-dimensional vectors. In order to compare with the "ground truth" solution obtained from the eigen-decomposition of the sample covariance matrix, the number of images is restricted to 90. This time, the connectivity between nodes is tested for $s = 2240$. For the stepsize parameters, we take $\eta_1 = 5 \cdot 10^{-8}$ for the first principal axis and a different one, $\eta_2 = 1 \cdot 10^{-7}$ for the second principal axis, in order to insure a better convergence. Note that $\eta_2 \geq \eta_1$ because the estimation of the second principal axis $\boldsymbol{w}_{2,t}$ requires the estimation of the first principal axis $\boldsymbol{w}_{1,t-1}$. The same stepsize values are taken for cooperative and noncooperative strategies. As for the IT algorithm, we take $\eta_1 = \eta_2 = 0.05$, for the Rayleigh quotient algorithm $\eta_1 = \eta_2 = 5 \cdot 10^{-6}$, for the OJAn algorithm $\eta_1 = \eta_2 = 5 \cdot 10^{-8}$, and for the LUO algorithm $\eta_1 = 5 \cdot 10^{-11}$ and $\eta_2 = 2.5 \cdot 10^{-10}$, in cooperative and noncooperative strategies.

Fig. 4 shows the convergence of the different orthogonalization processes using Oja's rule. It shows that the Gram-Schmidt orthogonalization and the deflation schemes have a comparable efficiency, especially with the cooperative strategies. The Gram-Schmidt orthogonalization is slightly better than the deflation scheme. In what follows, we use the Gram-Schmidt orthogonalization for multiple axes.

Fig. 5 to Fig. 7 show the convergence of each strategy for Oja's rule and for the IT, the Rayleigh quotient, the OJAn, and the LUO algorithms respectively, with the angles averaged over the N nodes. Again, the noncooperative strategy is significantly outperformed by all diffusion strategies. On the other hand, considering the rate of convergence, the figures show that the OJAn, the Oja, and the IT algorithms have the fastest convergence, followed consecutively by the RQ and the LUO algorithms. In terms of the floor level, the IT algorithm has the largest average angle at the convergence. All the other algorithms have a comparable level of the floor. As in the WSN application given in Section VII-A, the combination of the Oja or the OJAn algorithms with the adapt-then-combine diffusion strategy seems to present the best compromise between the angle floor and the rate of convergence.

VIII. Conclusion

In this paper, we proposed several strategies for estimating the principal axes for PCA in networks. Noncooperative and cooperative strategies with information diffusion were studied for this purpose. Experiments were conducted by taking into account the constraints imposed in WSNs and image processing applications. The results showed the relevance of these strategies. The convergence of all proposed algorithms was better than the non-cooperative strategy, in the case of WSNs. The main reason for this result is the ratio of the first and second eigenvalues which is greater in the case of wireless sensor networks. Furthermore, the level of efficiency of certain cost functions, like with the RQ and the LUO, highly depends on the choice of the initialization of the principal axes, which is not the case for the Oja and the OJAn rules. A similar behavior was observed for the image processing application. As a future work, we will also include the use of the spatial information within the study.

IX. Acknowledgement

This work is supported by the Région Champagne-Ardenne (grant "WiDiD") and the Lebanese University.

REFERENCES

[1] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[2] B. M. Wise, N. L. Ricker, and D. J. Veltkamp, "Upset and sensor failure detection in multivariable processes," AlChE Meeting, San Francisco, 1989.

[3] J. MacGregor, "Multivariate statistical methods for monitoring large data sets from chemical processes," AlChE Meeting, San Francisco, 1989.

[4] P. Honeine, "Online kernel principal component analysis: a reduced-order model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1814–1826, September 2012.

[5] N. Chitradevi, K. Baskaran, V. Palanisamy, and D. Aswini, "Designing an efficient PCA based data model for wireless sensor networks," in *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief*, ser. ACWR '11. New York, NY, USA: ACM, 2011, pp. 147–154.

[6] F. Chen, F. Wen, and H. Jia, "Algorithm of data compression based on multiple principal component analysis over the wsn," in *Wireless Communications Networking and Mobile Computing (WiCOM), 6th International Conference on*, Sept 2010, pp. 1–4.

[7] A. Rooshenas, H. Rabiee, A. Movaghar, and M. Naderi, "Reducing the data transmission in wireless sensor networks using the principal component analysis," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Sixth International Conference on*, Dec 2010, pp. 133–138.

[8] M. Ahmadi Livani and M. Abadi, "A PCA-based distributed approach for intrusion detection in wireless sensor networks," in *Computer Networks and Distributed Systems (CNDS), 2011 International Symposium on*, Feb 2011, pp. 55–60.

[9] L. Huang, M. I. Jordan, A. Joseph, M. Garofalakis, and N. Taft, "In-network pca and anomaly detection," in *NIPS*. MIT Press, 2006, pp. 617–624.

[10] J. R. Bunch and C. P. Nielsen, "Updating the singular value decomposition," *Numerische Mathematik*, vol. 31, pp. 111–129, 1978.

[11] P. M. Hall, A. D. Marshall, and R. R. Martin, "Merging and splitting eigenspace models." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 9, pp. 1042–1049, 2000.

[12] H. Kargupta, W. Huang, K. Sivakumar, B.-H. Park, and S. Wang, "Collective principal component analysis from distributed, heterogeneous data," in *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*. London, UK, UK: Springer-Verlag, 2000, pp. 452–457.

[13] Y.-A. Le Borgne, S. Raybaud, and G. Bontempi, "Distributed principal component analysis for wireless sensor networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008.

[14] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, vol. 15, no. 3, pp. 267–273, November 1982.

[15] N. Ghadban, P. Honeine, C. Francis, F. Mourad-Chehade, and J. Farah, "Strategies for principal component analysis in wireless sensor networks," in *Proc. eighth IEEE Sensor Array and Multichannel Signal Processing Workshop*, A Coruña, Spain, 22–25 June 2014.

[16] M. D. Plumbley, "Lyapunov functions for convergence of principal component algorithms," *Neural Networks*, vol. 8, pp. 11–23, 1995.

[17] G. Cirrincione, M. Cirrincione, J. Herault, and S. Van Huffel, "The mca exin neuron for the minor component analysis," *Neural Networks, IEEE Transactions on*, vol. 13, no. 1, pp. 160–187, Jan 2002.

[18] F.-L. Luo, R. Unbehauen, and A. Cichocki, "A minor component analysis algorithm," *Neural Networks*, vol. 10, no. 2, pp. 291 – 297, 1997.

[19] A. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *Signal Processing Magazine, IEEE*, vol. 30, no. 3, pp. 155–171, May 2013.

[20] R. Dony and S. Haykin, "Neural network approaches to image compression," *Proceedings of the IEEE*, vol. 83, no. 2, pp. 288–303, Feb 1995.

[21] J. Chen and A. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *Signal Processing, IEEE Transactions on*, vol. 60, no. 8, pp. 4289–4305, Aug 2012.

[22] M. H. Degroot, "Reaching a Consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[23] S.-Y. Tu and A. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *Signal Processing, IEEE Transactions on*, vol. 60, no. 12, pp. 6217–6234, Dec 2012.

[24] A. Nedic and A. Ozdaglar., "Cooperative distributed multi-agent optimization," in *Convex Optimization in Signal Processing and Communications*, D. P. Palomar and Y. C. Eldar, Eds. Cambridge University Press, 2009.

[25] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *arXiv preprint arXiv:1310.7063*, 2013.

[26] Y. Miao and Y. Hua, "Fast subspace tracking and neural network learning by a novel information criterion." *IEEE Transactions on Signal Processing*, vol. 46, no. 7, pp. 1967–1979, 1998.

[27] A. Taleb and G. Cirrincione, "Against the convergence of the minor component analysis neurons," *Neural Networks, IEEE Transactions on*, vol. 10, no. 1, pp. 207–210, Jan 1999.

[28] R. Möller, "First-order approximation of gram-schmidt orthonormalization beats deflation in coupled pca learning rules," *Neurocomput.*, vol. 69, no. 13-15, pp. 1582–1590, Aug. 2006.

[29] V. Srivastava, "A unified view of the orthogonalization methods," *Journal of Physics A: Mathematical and General*, vol. 33, no. 35, p. 6219, 2000.

[30] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network." *Neural Networks*, vol. 2, pp. 459–473, 1989.

[31] ——, "Two iterative algorithms for computing the singular value decomposition from input/output samples," in *Advances in Neural Information Processing Systems 6*, J. Cowan, G. Tesauro, and J. Alspector, Eds., 1994, pp. 144–151. [Online]. Available: http://papers.nips.cc/paper/869-two-iterative-algorithms-for-computing-the-singular-value-decomposition-from-inputoutput-samples.pdf

[32] C. Chatterjee, "Adaptive algorithms for first principal eigenvector computation." *Neural Networks*, vol. 18, no. 2, pp. 145–159, 2005.

[33] H. Kargupta, W. Huang, K. Sivakumar, B.-H. Park, and S. Wang, "Collective principal component analysis from distributed, heterogeneous data." in *PKDD*, ser. Lecture Notes in Computer Science, D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, Eds., vol. 1910. Springer, 2000, pp. 452–457.

[34] M. A. Livani and M. Abadi, "A PCA-based distributed approach for intrusion detection in wireless sensor networks," in *International Symposium on Computer Networks and Distributed Systems*, 2011.

[35] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, pp. 102–114, 2002.

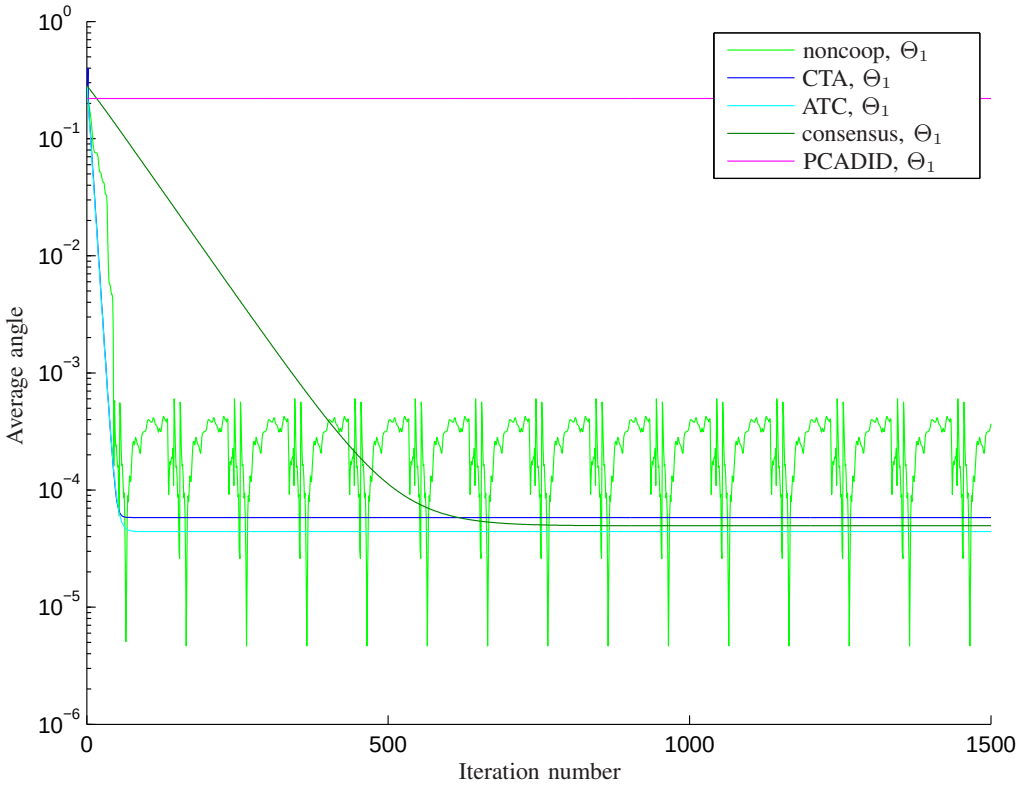

Fig. 2. Convergence analysis for Oja's rule, in the WSN settings.

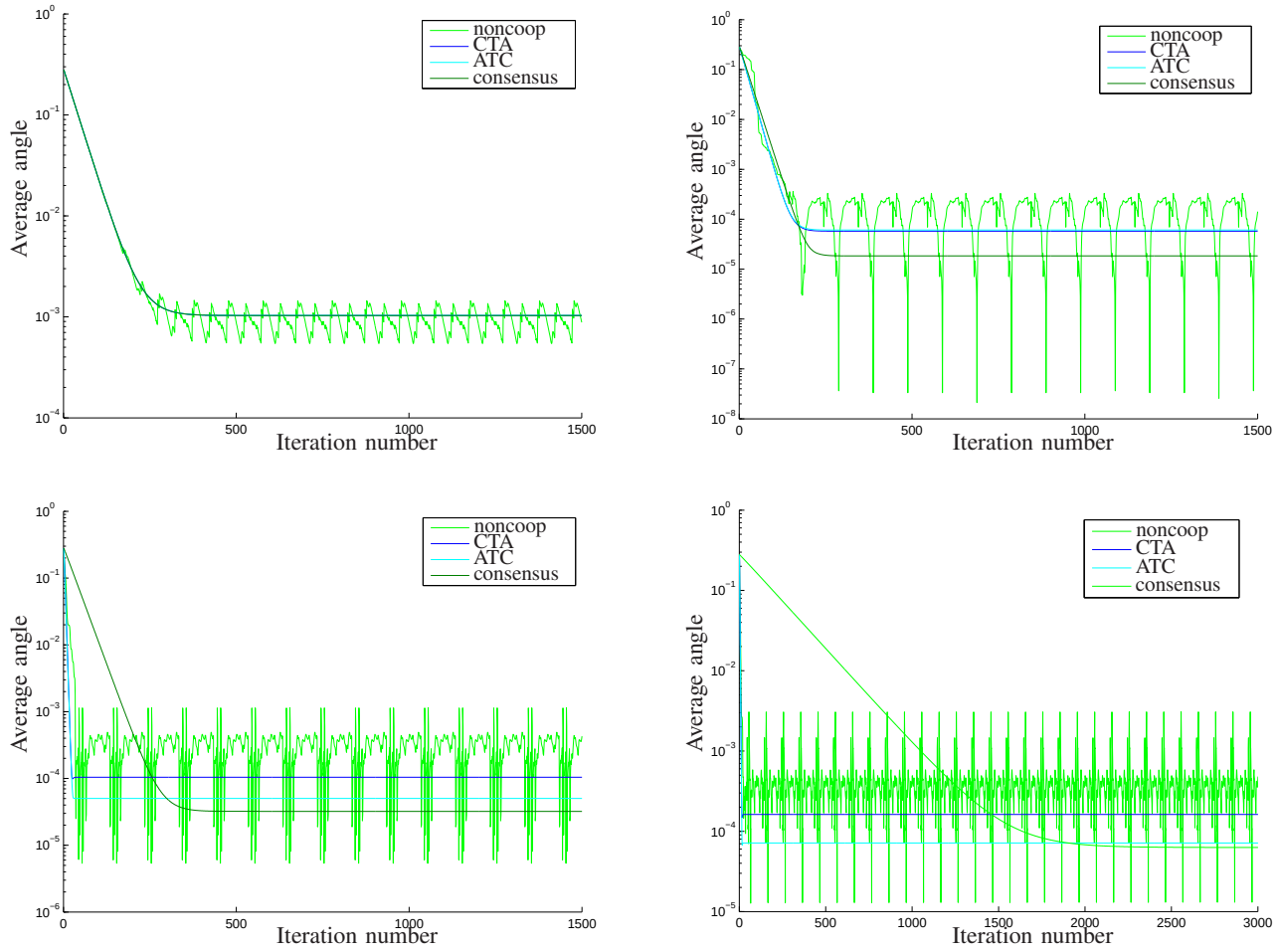

Fig. 3. Convergence analysis in the WSN settings of the strategies based on information theory (top left), on Rayleigh quotient (top right), on OJAn (bottom left) and on LUO (bottom right) algorithms.

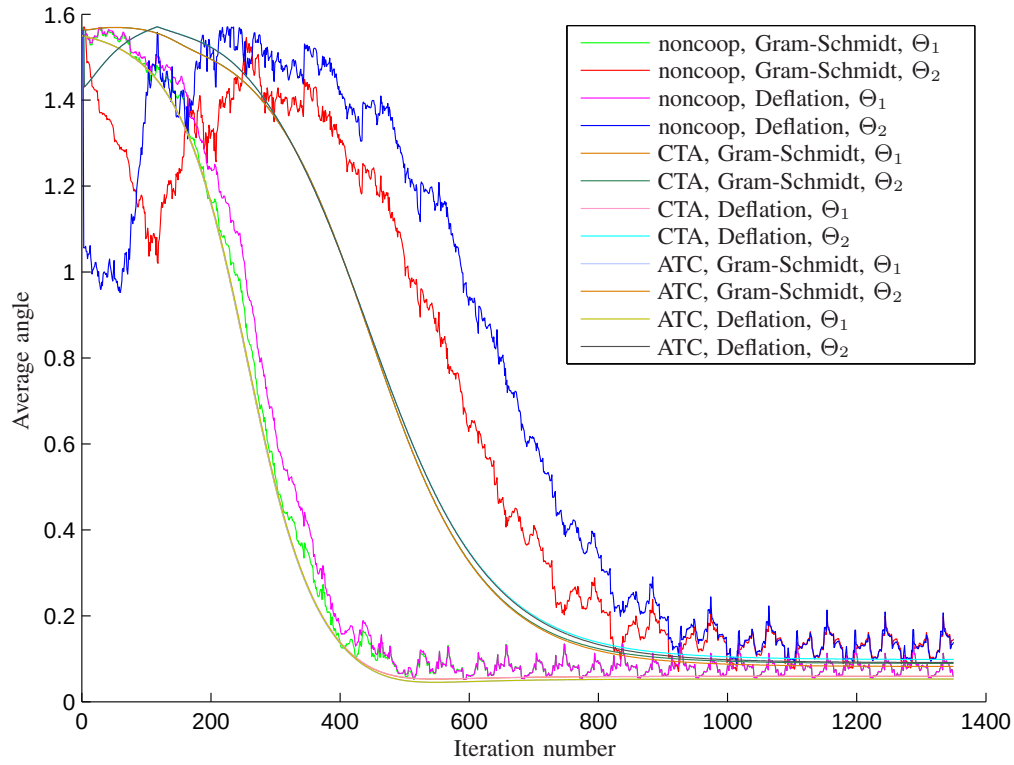

Fig. 4. Convergence analysis for the image dataset, with Oja's rule, for different orthogonalization processes.

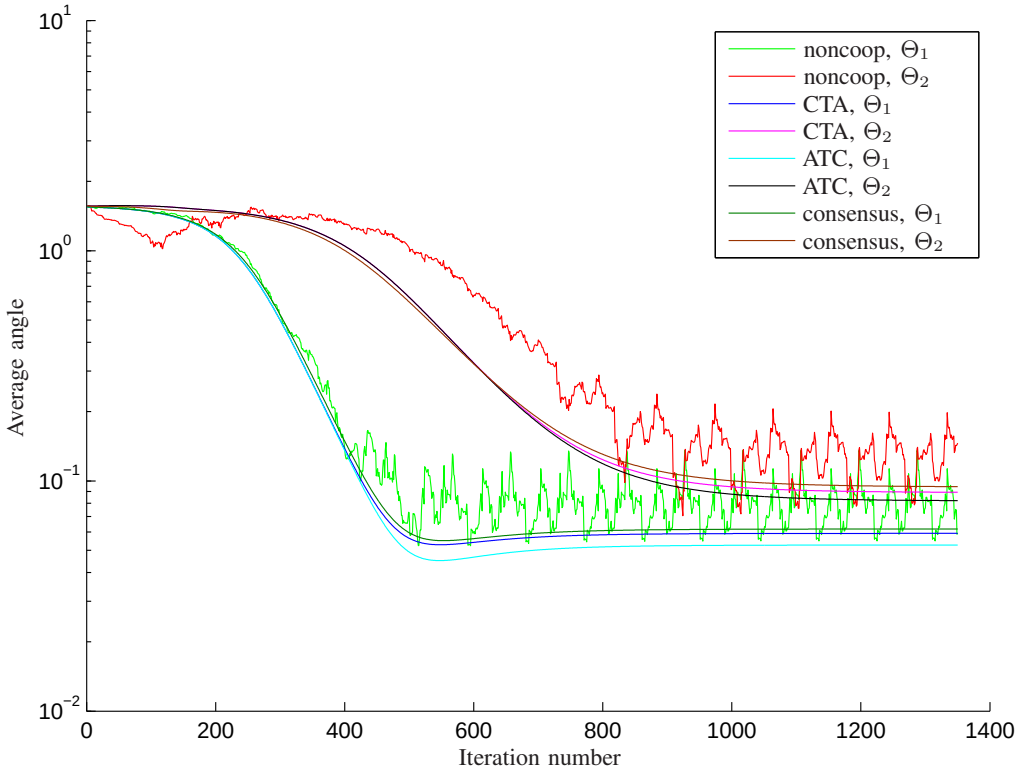

Fig. 5. Convergence analysis for the image dataset, with Oja's rule and Gram-Schmidt orthogonalization.

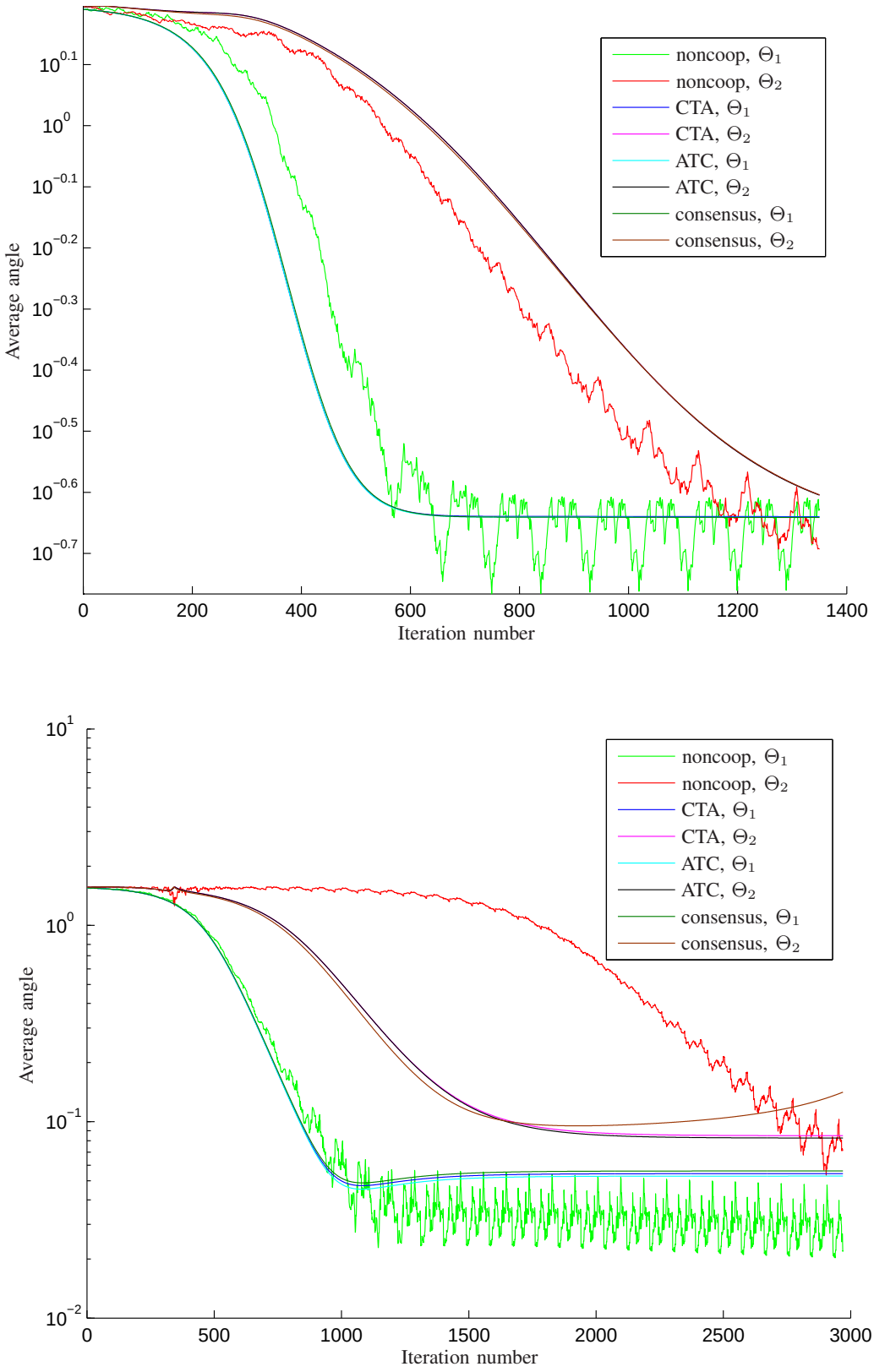

Fig. 6. Convergence analysis of the strategies based on the information theory (top figure) and the Rayleigh quotient (bottom figure), for the image dataset.

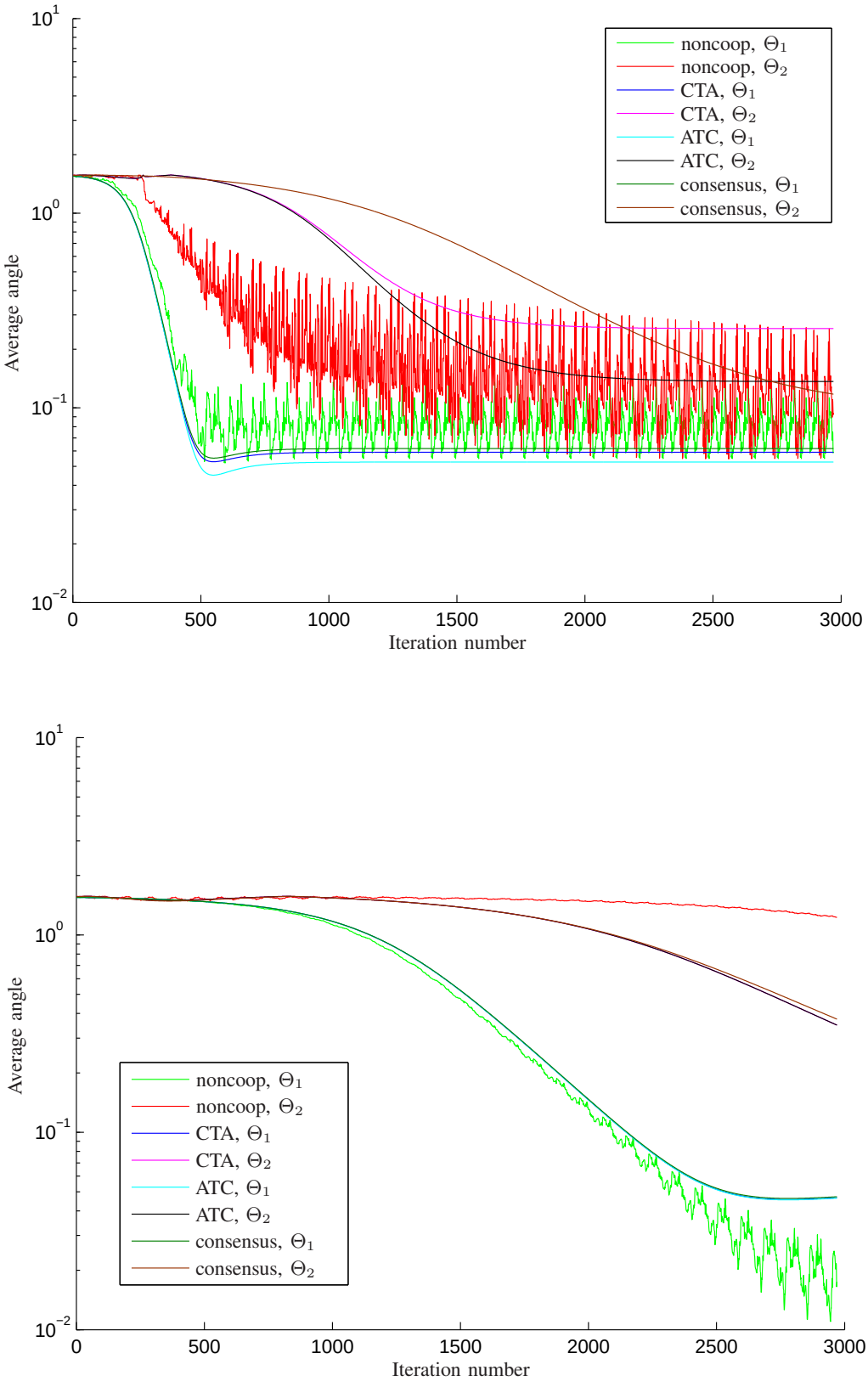

Fig. 7. Convergence analysis of the strategies based on the OJAn algorithm (top figure) and the LUO algorithm (bottom figure), for the image dataset.